Ethernet-based communication framework for sensor integration on industrial robots

M.W. de Graaf, R.G.K.M. Aarts, J. Meijer and J.B. Jonker
University of Twente - Laboratory of Mechanical Automation
P.O. Box 217 - 7500 AE Enschede - The Netherlands
m.w.degraaf@utwente.nl

***Abstract*— This paper presents a communication framework for integrating sensors in a robotic laser welding system. The framework was originally developed for the application of sensor-guided robotic laser welding, but the principles are generic, making it useful in other application areas as well. To use the sensor measurements during the robot motion, a synchronisation mechanism is presented for an image-based seam-tracking sensor that is integrated with an industrial 6-axis robot. The communication layer uses Ethernet TCP/IP communication and the synchronisation mechanism uses Ethernet UDP communication. Experimental validation is performed on industrial equipment to determine the accuracy of the proposed synchronisation mechanism.**

I. Introduction

Accurate path tracking of 3D trajectories using image-based sensors has received considerable attention in recent years. An application where this plays an important role is sensor-guided robotic laser welding. Laser welding requires a high accuracy of the position of the laser focal point with respect to the seam trajectory. Accuracies down to 0.1 mm have to be achieved at welding speeds up to 250 mm/s. Therefore seam-tracking sensors, that measure the seam trajectory close to the laser focal spot, are required [1]. To integrate such sensors in a robotic system, they need to communicate with the robot controller. Furthermore, time delays that occur and synchronisation issues need proper attention.

Two strategies are distinguished for sensor-guided robotic laser welding:

1) Teaching of the seam trajectory with the sensor in a first step (seam teaching) and laser welding in a second step.
2) Real-time seam tracking during laser welding. When the sensor is mounted some distance in front of the laser focal point it can measure the seam trajectory and let the laser focal point track the measured trajectory in the same movement.

Seam teaching (method 1) has the advantage that it can be performed using point-to-point movements, where the robot stabilises after every step. This increases the accuracy as dynamic robot behaviour and synchronisation errors can be avoided.

In a production environment method 2 would be preferred as it is faster and thus saves time and money. The seam-tracking sensor measures relative to its own coordinate frame (which is attached to the end-effector of the robot arm). Therefore its measurements can not be directly used during a robot movement. The sensor measurements will only be useful if the Cartesian location of the sensor frame in the robot workspace (computed from the robot joint angles) is known at the same time. This can be accomplished in two ways:

- Let the robot make a movement and wait until it stabilises. Because the robot is stabilised the location of the sensor frame in the robot workspace does not change in time. Then, a sensor measurement can be easily related to the corresponding robot position.
- When a sensor measurement is obtained during the robot motion, the time axes of the robot and the sensor need to be synchronised. If these are synchronised the robot joints can be interpolated to match the sensor measurement with the robot joints or vice versa.

Robot-sensor synchronisation can easily be achieved for a simple external sensor (like a LVDT), where the sensor operates continuously and is interfaced directly to the robot, because the sensor data is immediately available when the robot needs it. Synchronisation becomes more challenging when complex sensor systems, like camera-based sensors with their own real-time clock are used. This paper addresses the synchronisation for such sensors.

The paper describes a communication framework between a robot controller, a sensor and a user application. Section II describes how typical components are connected. The communication framework is described in section III. Section IV gives an overview of the coordinate frames and transformations in a sensor-guided robotic laser welding system. Section V describes the synchronisation mechanism that is used to synchronise the robot joint measurements with the image acquisition of the seam-tracking sensor. Experiments have been performed on industrial equipment to determine the time delay of the synchronisation procedure (section VI).

II. Connection topology

Common components in a sensor-guided robotic laser welding system are a robot controller, seam-tracking sensor, laser source and a fiber to guide the laser light to a welding head, where it is focussed on the product. Process gas protects the melt pool from oxidation and a crossjet protects the focussing lens in the welding head from process spatters. This section shows how the components are connected and how they interact. The connection topology of the system is shown in figure 1.

ROBOCOMM 2007, 14th – 16th Oct 2007, Athens, Greece.
Copyright © 2011 – 2012 ICST
ISBN 978-963-9799-08-0
DOI 10.4108/ICST.ROBOCOMM2007. 2089

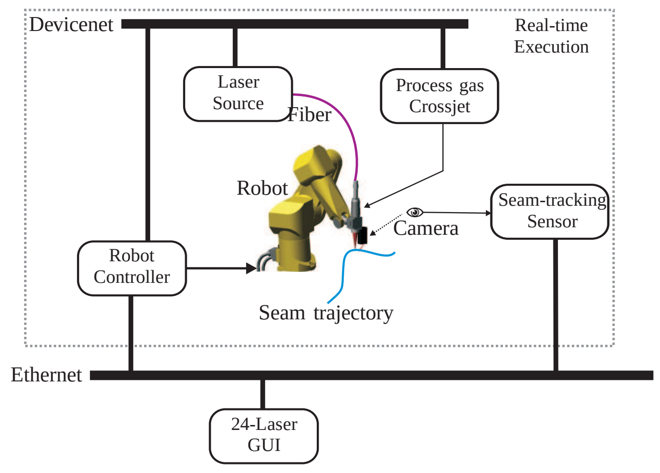


Fig. 1. Typical connection topology in a sensor-guided robotic laser welding system

All the components inside the dotted box require real-time execution. The robot controller not only controls the robot motion but also the other equipment during laser welding. The laser, crossjet and process gas have to be switched on and off synchronously with the robot motion. An industrial Devicenet bus system is used for this purpose. The seam-tracking sensor is also located within the real-time execution box, as its measurements must be synchronised with the measurements of the robot joint angles. The synchronisation procedure uses Ethernet UDP-communication and is explained in section V.

The operator can control the system using the 24-Laser graphical user interface (GUI), which does not require real-time guarantees. 24-Laser is located on a computer that is connected to the robot controller using Ethernet. It is used to prepare and process a laser welding job. Furthermore it contains the implementations of tool calibration procedures [2], [3], seam teaching algorithms [4] and seam tracking algorithms [5].

III. Communication framework

The software architecture is schematically shown in figure 2. The software that was developed in this work is located on three different computer systems: The robot controller, the sensor system and the PC for the 24-Laser GUI. The software is mainly written in C++.

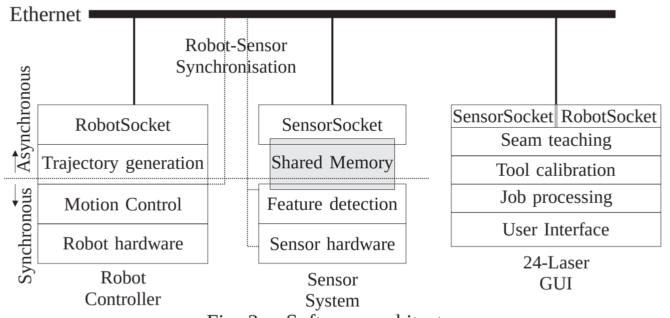


Fig. 2. Software architecture

On the robot controller, the basic joint motion control is performed in the Motion Control layer, which is developed by Stäubli [6]. The Trajectory Generation layer computes smooth reference setpoints for the Motion controller at fixed times. Using this trajectory generator, the robot can be moved in various modes (joint motion, Cartesian motion, etc). The RobotSocket server is the communication layer that allows clients to connect to the robot and use it (section III-A).

The image-processing of the seam-tracking sensor is performed in the Image Processing layer. After a CMOS image has been processed, the extracted features are stored in a piece of shared memory. The extracted features are communicated to clients that connect to the SensorSocket server (section III-B).

A user application can access the robot and sensor through a RobotSocket client and a SensorSocket client respectively. The 24-Laser GUI is an example of a user application that is especially suited for sensor-guided robotic laser welding. For other application areas, matching user applications can be programmed. The communication between the robot, sensor and user application is generic, different robots and sensors providing the correct Application Programming Interface (API) can be used as well.

A. *RobotSocket*

The RobotSocket communication layer is an Ethernet socket communication layer to let the robot communicate with its surroundings. It uses the client-server model: on the side of the robot controller, a RobotSocket server waits for RobotSocket clients that connect to it. The RobotSocket communication layer is programmed as a C++ class.

The RobotSocket server listens on a specific port (default 4000) for incoming TCP request packets. Request packets need to have a certain structure in which e.g. a Command ID is included to specify the requested robot command. If the request packet is identified as being a correct request packet, the RobotSocket server performs the request based on the Command ID. If the request was performed correctly, the RobotSocket server responds with a reply packet. If an error occurs during processing of the request the RobotSocket server responds with an error packet.

This approach has several major advantages:

- Ethernet communication is fast (100 Mbit/s) and cheap. No additional I/O hardware is needed as the robot controller and the computers are standardly equipped with a network controller.
- Several error-checks are performed, which guarantees correct packet delivery and prevents unexpected robot behaviour.
- The system is robot-independent. Robots from different manufacturers and with different controllers can be used as long as they follow the specifications in the API as defined in the RobotSocket class.
- Robot emulators can be used. Instead of real robot hardware it is possible to connect to a robot emulator, which follows the RobotSocket API. This is ideal for testing and debugging purposes as no ”damage” can

occur. Furthermore, these robot emulators can contain models of the robot and its surrounding. This way the influence of different factors (kinematics, dynamics, tool transformation errors,etc) on the path accuracy can be predicted and visualised.

The API of the RobotSocket communication layer is developed in a generic way, so different commands using a list of different parameters can be sent and received. It can be easily extended with different commands by adding additional command ID's and parameters.

The commands that are communicated in the RobotSocket communication layer are similar to commands that can be found in common robot programming languages (VAL3 [7], V+, etc.). The RobotSocket API therefore provides remote robot control from a user application that is located somewhere on the network or even the Internet. Examples of commands are:

- joints = Herej(). Returns the current joint position.
- loc = Here(tool, frame). Returns the current Cartesian position and orientation of tool relative to frame.
- EnablePower(), DisablePower(). Enable or disable the robot power.
- JointMove(joints). Perform a joint-interpolated movement to the position in joints.
- ToolMovej(loc, tool, frame). Perform a joint-interpolated movement from the current robot position to the Cartesian location specified as tool relative to frame.
- ToolMovel(loc, tool, frame). Perform a Cartesian-interpolated movement from the current robot position to the Cartesian location specified as tool relative to frame.

B. *Sensorsocket*

The SensorSocket communication layer is an Ethernet socket communication layer, used to let the sensor communicate with its surroundings. It is similar to the RobotSocket communication layer, but now the SensorSocket server listens on a default port 2000 for incoming TCP request packets.

Examples of commands are:

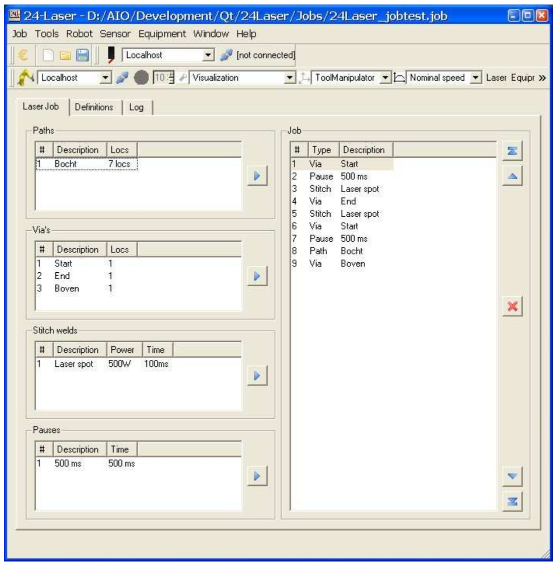
- data = GetData(). Returns the position and orientation of a seam location relative to the sensor frame.
- image = GetImage(). Returns the complete sensor image.

C. *24-Laser*

The 24-Laser GUI is the link between the robot operator and the hardware and software. 24-Laser allows the robot operator to perform a number of different tasks:

- Robot status and motion control. The status of the robot can be visualised, e.g. the status of Emergency Stops, Motion, Robot Power, Joint and Cartesian position. Furthermore, motion commands can be given to the robot.
- Seam teaching using point-to-point movements. Both manual teaching of locations and 3D seam teaching with the aid of a seam tracking sensor can be performed from the GUI.
- Tool Calibration using point-to-point movements. Automatic tool calibration procedures for finding the transformations of the laser tool and sensor tool can be performed from the GUI.
- Job preparation. This task consists of setting the parameters on a welding trajectory, e.g. velocity, laser on/off, laser power, process gas on/off, crossjet on/off.
- Job processing. A welding job may consist of paths (welding trajectories), via locations (in-between locations), stitch welds (spots) and pauses. During job processing these are processed in a user-defined order.

Furthermore the GUI contains a database where different tool transformations and frame transformations can be stored and edited. Welding jobs are loaded and saved in an XML file-format [8].



Paths

#	Description	Locs
1	Bocht	7 locs

Via's

#	Description	Locs
1	Start	1
2	End	1
3	Boven	1

Stitch welds

#	Description	Power	Time
1	Laser spot	500W	100ms

Pauses

#	Description	Time
1	500 ms	500 ms

Job

#	Type	Description
1	Via	Start
2	Pause	500 ms
3	Stitch	Laser spot
4	Via	End
5	Stitch	Laser spot
6	Via	Start
7	Pause	500 ms
8	Path	Bocht
9	Via	Boven

Fig. 3. Screenshot of the 24-Laser main window.

The 24-Laser GUI is developed using Trolltech's Qt software [9]. Qt is a C++ software framework used for programming Graphical User Interfaces. It features cross-platform capabilities as the source can be compiled for MS Windows, Linux and Macintosh operating systems. A 24-Laser screenshot is shown in figure 3.

24-Laser uses the RobotSocket and SensorSocket communication libraries that were described in the previous sections. Furthermore a number of dynamic libraries have been developed for use in 24-Laser:

- Seamtracking DLL. Used for real-time Seam-tracking. Details of the used algorithms can be found in [5].
- LaserCalib DLL. Used for laser tool calibration. The used algorithms and measurement method are described in [2].
- SensorCalib DLL. Used for sensor tool calibration. The used algorithms and measurement method are described in [3].
- ILC DLL. Used for Iterative Learning Control (ILC), which is a control strategy used to increase the tracking accuracy of the joint motion controller, by repeating the

trajectory and learning from errors made in previous runs. ILC is described in [10].

As 24-Laser can communicate with the robot controller and sensor it can combine measured seam positions with the current robot configuration to correct for tracking errors as will be addressed in the next section.

IV. Coordinate frames and transformations

To describe the position and orientation of points or bodies a coordinate system or frame is attached to each body. Frames are indicated by a bold capital. A transformation describes the location (position and orientation) of a frame with respect to a reference frame. Transformations are indicated by the symbol $\mathbf{T}$ with a leading superscript, that defines the reference frame they refer to. The leading subscript defines the frame they describe, e.g. transformation ${}^{A}_{B}\mathbf{T}$ describes frame $\mathbf{B}$ with respect to frame $\mathbf{A}$.

In literature, a 4x4 homogenous transformation matrix is often used to describe a transformation. It consists of a 3x3 rotation matrix and 3D position vector. A transformation ${}^{A}_{B}\mathbf{T}$ can be written as

$$ {}^{A}_{B}\mathbf{T} = \begin{bmatrix} {}^{A}_{B}\mathbf{R} & {}^{A}_{B}\mathbf{P} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{1} $$

where ${}^{A}_{B}\mathbf{R}$ is the rotation matrix that describes the orientation of frame $\mathbf{B}$ with respect to frame $\mathbf{A}$ and ${}^{A}_{B}\mathbf{P}$ is the position vector that describes the position of frame $\mathbf{B}$ with respect to frame $\mathbf{A}$. Homogenous transformation matrices have some useful properties, e.g. different transformations can be added using matrix multiplication.

An overview of the different frames and transformations that are used in this work is given in figure 4. A more generic description can be found in De Graaf et al [4].

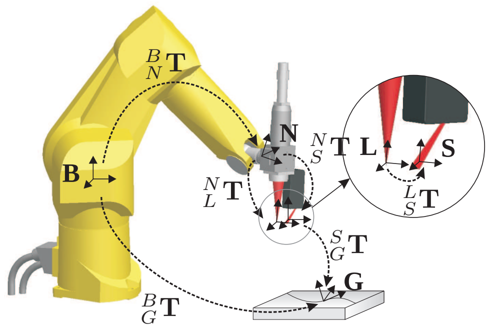


Fig. 4. Frames in a sensor-guided robotic laser welding system

The following frames are distinguished:

- Base or world frame $\mathbf{B}$. This frame is attached to the robot base and does not move with respect to the environment.
- Null or flange frame $\mathbf{N}$. The null frame is located at the end of the robot flange. The null frame is described with respect to $\mathbf{B}$ by coordinate transformation ${}^{B}_{N}\mathbf{T}$, which is a function of the joint angles of the robot arm.
- Laser tool frame $\mathbf{L}$. The Laser tool is located at the Tool Center Point of the focussed laser beam, where the z-axis of this frame coincides with the laser beam axis. Because the laser beam is axi-symmetric, the direction of the x-axis is arbitrary. Its direction will be chosen as close as possible towards the Sensor tool. The transformation ${}^{N}_{L}\mathbf{T}$ describes the laser tool frame with respect to the Null frame. This is a fixed transformation determined by the geometry of the welding head.
- Sensor tool frame $\mathbf{S}$. The seam tracking sensor is fixed to the welding head and therefore indirectly to the robot flange. The transformation ${}^{N}_{S}\mathbf{T}$ describes the sensor tool frame with respect to the Null frame, where the x-axis of $\mathbf{S}$ is chosen in the welding direction. Note that, because both transformations are fixed, this transformation can also be described with respect to the laser tool frame instead of the null tool by transformation ${}^{L}_{S}\mathbf{T}$.
- Robot tool frame $\mathbf{T}$. A robot movement is more generally specified by the movement of a robotic tool frame $\mathbf{T}$, which can be the sensor tool $\mathbf{S}$ or laser tool $\mathbf{L}$.
- Seam frame $\mathbf{G}$. Every location on a seam trajectory is described with a different coordinate frame. The transformation ${}^{B}_{G}\mathbf{T}$ describes a seam frame with respect to the base frame.

If the seam trajectory is within the field-of-view of the seam-tracking sensor, it measures a location ${}^{S}_{G}\mathbf{T}$ on the seam trajectory with respect to its own coordinate frame $\mathbf{S}$. A seam location with respect to the fixed base frame $\mathbf{B}$ can be calculated by following the chain of transformations as

$$ {}^{B}_{G}\mathbf{T} = {}^{B}_{N}\mathbf{T} \cdot {}^{N}_{S}\mathbf{T} \cdot {}^{S}_{G}\mathbf{T}. \tag{2} $$

Transformation ${}^{B}_{N}\mathbf{T}$ represents the location of the robot flange with respect to the base frame. It is calculated from the measured robot joint angles $\mathbf{q}_m$ as

$$ {}^{B}_{N}\mathbf{T} = DGM(\mathbf{q}_m), \tag{3} $$

where $DGM(\mathbf{q}_m)$ depends on the geometry of the robot arm (e.g. arm lengths, encoder offsets). In literature $DGM(\mathbf{q}_m)$ is often called the Direct Geometric Model [11] or forward kinematics function [12], [13]. Khalil and Dombre have derived symbolic expressions for equation 3 for the specific case of a 6-axes Stäubli robot.

Transformation ${}^{S}_{G}\mathbf{T}$ is computed from a sensor image using image processing. If the robot is moving, both these transformations change in time. Therefore to accurately compute a seam location ${}^{B}_{G}\mathbf{T}$, the robot joint measurements need to be synchronised with the sensor image acquisition.

V. Synchronisation procedure

An important topic in a sensor-guided robotic laser welding system is the synchronisation between the robot controller and the sensor. In figure 5 the used synchronisation procedure is shown. The robot controller and the seam-tracking sensor have their own time axes. The measured robot joint angles are available at fixed time intervals T_r (4 ms in our case). The times $t^R_k = k \cdot T_r$ and corresponding robot joint measurements $\mathbf{q}^{\mathrm{m}}_k$ are stored in a cyclic buffer on the robot controller.

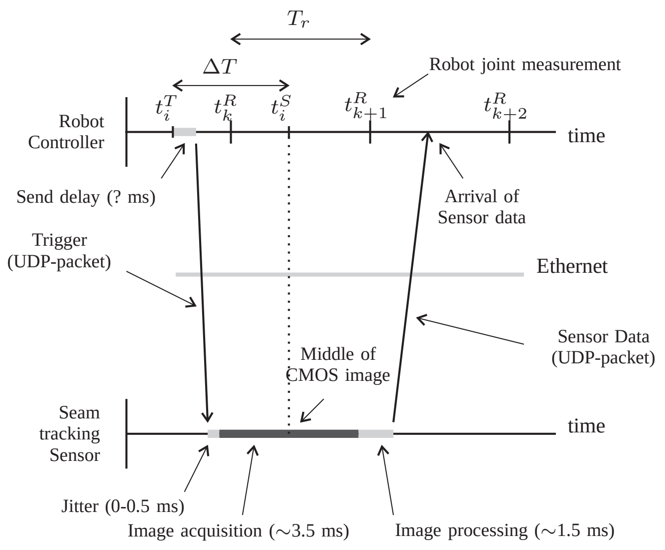


Fig. 5. Synchronisation method

The acquisition of a single CMOS image is triggered, by sending a trigger UDP (User Datagram Protocol) packet to the sensor. The robot controller sends these trigger packets at times t_i^T, where i is the trigger packet index. On arrival of a trigger packet at the sensor, the image acquisition is started. The image acquisition takes (depending on the field-of-view) a certain time. The image processing time depends strongly on the chosen feature detection algorithm and CPU that is used for image processing. After the image processing has been completed, the sensor data ${}^S_G\mathbf{T}_i$ is transmitted to the robot controller. Both the trigger packet and the sensor data packet contain the index i. So once a sensor data packet arrives at the robot controller, the corresponding t_i^T is known.

During the robot motion, the sensor is normally positioned symmetrically with respect to the seam, implying that the seam point is located somewhere in the middle of the CMOS image. Since the CMOS chip is read column-wise from one side to the other, detection of the seam point will take place about half-way during the image acquisition. In figure 5 it can be seen that the estimated time t_i^S at which the sensor measurement took place is computed as

$$t_i^S = t_i^T + \Delta T, \tag{4}$$

where t_i^T is the time at which the trigger packet was sent to the sensor and ΔT is a time delay. In section VI experiments are described to determine ΔT.

Let $\mathbf{q}_k^{\mathrm{m}}$ and $\mathbf{q}_{k+1}^{\mathrm{m}}$ be robot joint measurements at times t_k^R and t_{k+1}^R respectively and t_i^S lies in the interval between t_k^R and t_{k+1}^R. Then the robot joint angles $\mathbf{q}^{\mathrm{m}}(t_i^S)$ can be approximated using linear interpolation as

$$\mathbf{q}^{\mathrm{m}}(t_i^S) = \frac{(t_i^S - t_k^R)\mathbf{q}_k^{\mathrm{m}} + (t_{k+1}^R - t_i^S)\mathbf{q}_{k+1}^{\mathrm{m}}}{t_{k+1}^R - t_k^R}. \tag{5}$$

The synchronisation procedure is summarised as follows:

1) The robot controller sends trigger packets to the sensor at times t_i^T.
2) On arrival of the sensor data at the robot controller, the estimated time t_i^S when the image acquisition took place is calculated using equation 4.
3) Find the interval, where t_i^S lies between t_k^R and t_{k+1}^R.
4) Compute the interpolated joint angles $\mathbf{q}^{\mathrm{m}}(t_i^S)$ using equation 5.
5) A seam location ${}^F_G\mathbf{T}$ can be calculated from the interpolated joint angles and the received sensor data using equations 2 and 3. Only the order of incoming seam locations needs to be known to construct the seam trajectory.

The total time delay ΔT depends on the equipment used, but is expected to be constant with a certain amount of jitter (variation in time delay). In our case, ΔT consists of half the image acquisition time, which takes (depending on the field-of-view) about $\frac{3.5}{2} = 1.75$ ms for a full frame of 512 x 256 pixels. There is a communication delay, which is in the order of 0.1 ms. Furthermore, ΔT also accounts for the fact that the times t_k^R at which the joint measurements are available at the robot controller are different from the actual time that the encoders are measured. Because ΔT is experimentally determined, this delay (and others that may exist) are also taken into account.

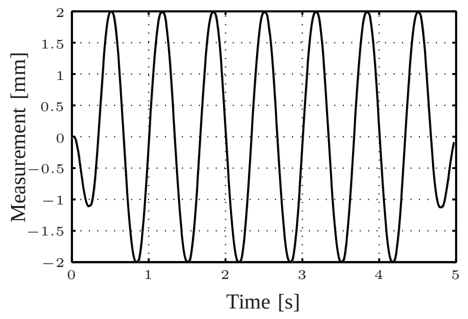
The sensor checks at a rate of 4 kHz whether a trigger packet has arrived, which gives a jitter of 0.25 ms. There is another jitter of 0.25 ms before the image acquisition actually starts. Therefore, the total jitter between receiving the trigger packet and the start of the image acquisition is 0.5 ms. Although the use of the UDP-protocol on a switched network does not guarantee a fixed time delivery of packets on the network, packet delivery time is low (less than 0.1 ms) compared to the image acquisition time for a moderate network load and therefore the jitter caused by the UDP communication is also less than 0.1 ms. The total jitter in ΔT is therefore about 0.6 ms.

The presented synchronisation procedure has the following advantages:

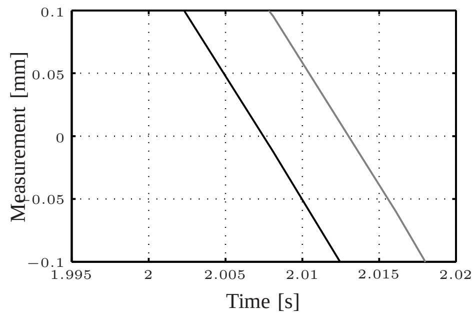
- Only the arrival of the trigger packet is important, which makes the procedure independent of a varying image processing time after the image acquisition.
- The low jitter in the system makes it very suitable for the high accuracy requirements of laser welding
- The trigger packets can be sent completely independently of the robot sample time. It is not necessary to sent them at fixed time intervals.
- The interpolation is calculated after arrival of the sensor data at the robot controller. If a packet does not arrive (e.g. due to a communication error) it is automatically ignored.

VI. Measuring the time delay

To measure the time delay ΔT, the following experiment has been performed. An object with a straight seam has been put in the middle of the field-of-view of the sensor. The sensor



(a) Complete motion



(b) Zoomed around 2 s

Fig. 6. Measured displacement of a sine motion with a frequency of 1.5 Hertz (— $y^R(t_k^R)$, — $y^S(t_i^T)$)

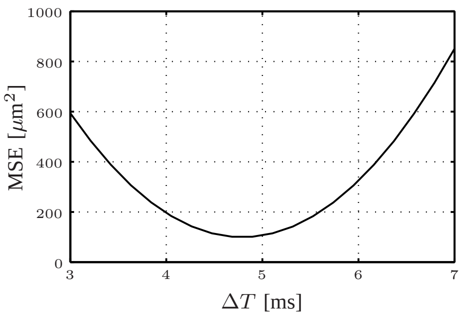


Fig. 7. Mean-squared error as a function of the delay time for a sine motion with a frequency of 1.5 Hertz

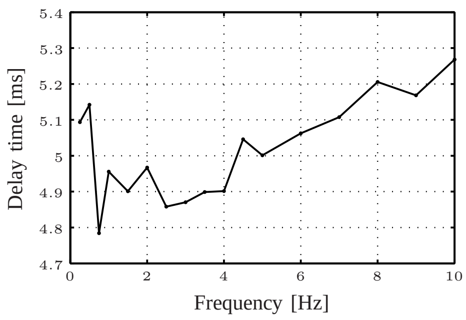


Fig. 8. Computed delay time as a function of the frequency of the sine motion

is moved perpendicular to the seam trajectory using a sine-motion with an amplitude of 2 mm. The measured displacements y^R of the sensor tool frame should now correspond to the measured displacements y^S of the sensor. The sensor readings y^S are recorded with respect to the times t_i^T and the sensor tool displacements y^R (calculated from the robot joints measurements) are recorded with respect to the times t_k^R. In figure 6(a), the result of this measurement for a frequency of 1.5 Hz is shown. The time delay between the two sine-measurements should correspond to the time delay ΔT. As expected both measurements closely fit. To see the time delay figure 6(b) shows the same measurement, but is zoomed in around 2 s.

The sensor measurements $y^S(t_i^T)$ are known at times t_i^T and the robot measurements $y^R(t_k^R)$ are known at times t_k^R. To be able to compare them, they need to be known with respect to a common time or index. Therefore, both the sensor measurements and robot measurements are upsampled to 5 kHz. The common times are denoted by $t_j = j \cdot 0.2$ ms. The residual errors $e(t_j, \Delta T)$ are defined as

$$e(t_j, \Delta T)^2 = \left(y^S(t_j + \Delta T) - y^R(t_j)\right)^2, \tag{6}$$

where $y^S(t_j)$ and $y^R(t_j)$ are the upsampled sensor and robot measurements respectively and ΔT is a variable time delay. The mean-squared error MSE is a function of ΔT and is defined as

$$\text{MSE}(\Delta T) = \frac{1}{n} \sum_{j=1}^{n} e(t_j, \Delta T)^2, \tag{7}$$

where n is the total number of measurements. The value of ΔT is sought, where the mean-squared error MSE has a minimum. For the sine motion at 1.5 Hertz the MSE as a function of ΔT is given in figure 7.

The value of ΔT where the MSE has a minimum is found with a resolution of 0.2 ms. To further increase the resolution a second order polynomial has been fitted through the MSE values to accurately find the minimum. For the sine motion at 1.5 Hertz, the time delay is found to be 4.9 ms using this method.

It should be noted that ΔT can only be correctly calculated if the robot measurements at the joint level correspond with the sensor measurement at the robot tip. This is the case when the robot accurately tracks the reference sine motion, but does not apply when the robot shows flexible behaviour of the joints and links, i.e. at high frequencies. On the other hand, at low frequencies, the signal-to-noise ratio is expected to be low as actually a phase shift is measured, which can not be interpreted accurately as a time delay at these frequencies. To see these effects, the time delay is calculated for sine motions at different frequencies ranging from 0.1 Hertz to 10 Hertz. In figure 8, the time delay is plotted as a function of the frequency.

An almost constant time delay is found for frequencies in the range of 1 Hertz to 4 Hertz. If the sine frequency is increased the delay time increases due to elastic robot behaviour. At low sine-frequencies the computed delay time becomes inaccurate due to the lower signal-to-noise ratio.

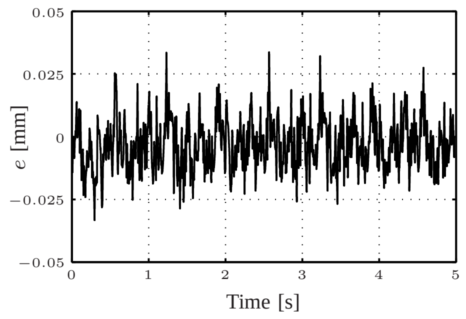


Fig. 9. Residual error of the sine motion at 1.5 Hertz after synchronisation

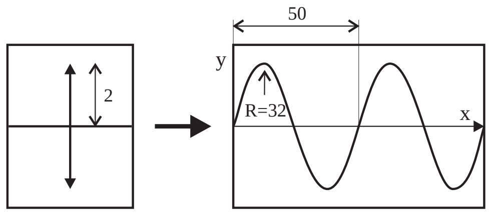


Fig. 10. Static sine motion translated to curved sine trajectory

To show the accuracy of the synchronisation method, a delay time of $\Delta T = 4.9$ ms is used in the synchronisation procedure. A sine motion at 1.5 Hertz is carried out and the residues between the synchronised robot tip measurement and sensor measurement have been plotted in figure 9.

The maximum value of the residues is about 30 μm, which is in the same order as the robot repeatability. No remaining sine wave at 1.5 Hz can be observed, which shows that the delay time is correctly applied.

The sine motions were performed on a static location on the seam trajectory. The accuracies that were obtained in these experiments can be translated to accuracies that occur during measurements of curved seam trajectories with a robot that is moving at typical laser welding speeds (figure 10).

For a sine motion of 5 Hertz with an amplitude of 2 mm, the residues at this frequency are about 40 μm, which is well below the desired accuracy for laser welding. Suppose this sine motion was performed at a linear velocity of 250 mm/s, then in one period a distance x of 50 mm would have been travelled. The distance y perpendicular to the welding direction x can then be described as

$$y(x) = 2 \sin \frac{5 \cdot 2\pi \cdot x}{250}. \tag{8}$$

The radius of curvature R for a curve written in the form $y = f(x)$ [14] can be computed using

$$R = \frac{[1 + \frac{dy}{dx}^2]^{\frac{3}{2}}}{|\frac{d^2y}{dx^2}|}. \tag{9}$$

The minimum radius of curvature is at the top (or bottom) of the sine, at x=12.5 mm or x=37.5 mm. It is computed as $R \approx 32$ mm. Suppose a synchronisation accuracy of 40 μm is desired at a welding speed of 250 mm/s, the radius of curvature should be larger than this. The computed radius of curvature is much better than the radius of curvature (typical value of 200 mm) that the robot can achieve, provided that the dynamic tracking accuracy stays in the order of 40 μm at these speeds. For lower welding speeds the radius of curvature that can be accurately measured is even better, which shows the accuracy of the proposed synchronisation method is more than sufficient for robotic laser welding.

VII. Discussion

In this paper, an Ethernet-based communication framework has been described for sensor-guided robotic laser welding. Although the framework aims at the application of laser welding, its principles are generic, making it useful for different application areas where sensors and robots are used. A well-defined extendible interface has been presented for asynchronous high-level commands (Ethernet TCP) between robots, sensors and a user application, which allows robots and sensor from different manufacturers to be used through the same interface.

To use sensor measurements in real-time during the robot motion, a synchronisation method (Ethernet UDP) is presented. Experiments are performed to determine the time-delay between the robot joint measurements and the sensor measurements, which is found to be 4.9 ms in our particular case. The jitter in the system is below 0.5 ms, which makes the synchronisation method very suitable for laser welding.

References

[1] J. P. Huissoon. Robotic laser welding: seam sensor and laser focal frame registration. *Robotica*, 20:261–268, 2002.

[2] J. Van Tienhoven. Automatic tool centre point calibration for robotised laserwelding. Master's thesis, University of Twente, 2003. Internal report WA-919.

[3] F. van der Velde. Development of an automatic calibration procedure for the sensor tcp. Master's thesis, University of Twente, June 2005. Internal report WA-989.

[4] M. W. De Graaf, R. G. K. M. Aarts, J. Meijer, and J. B. Jonker. Modeling the seam teaching process for robotic laser welding. In Paul Drews, editor, *Proceedings of the Mechatronics & Robotics 2004 conference*. APS - European Centre for Mechatronics, September 2004.

[5] M.B. Rensen. Development and implementation of a real-time seam tracking algorithm. Master's thesis, University of Twente, 2006. Internal report WA-1044.

[6] F. Pertin and J. M. Bonnet-des-Tuves. Real time robot controller abstraction layer. In *Proceedings of the International Symposium on Robotics 2004*, 2004.

[7] Stäubli. *VAL3 Reference Manual version 4.0*. Stäubli, Faverges, 2004.

[8] XML. Extensible markup language (XML), 2003.

[9] Trolltech. Qt - gui software toolkit, 2005.

[10] W. B. J. Hakvoort, R. G. K. M. Aarts, J. Van Dijk, and J. B. Jonker. Iterative learning control for improved end-effector accuracy of an industrial robot. In *Proceedings of Symposium on Robot Control*. IFAC, September 2006.

[11] W. Khalil and E. Dombre. *Modeling, Identification & Control of Robots*. Hermes Penton Ltd, 2002.

[12] J. J. Craig. *Introduction to robotics: Mechanics & Control*. Addison-Wesley publishing company, Inc., 1986.

[13] Richard P. Paul. *Robot manipulators: mathematics, programming and control*. MIT Press, Cambridge, MA, USA, 1982.

[14] E. Kreyszig. *Differential Geometry*. New York: Dover, 1991.